REMARKS

The Non-Final Office Action mailed July 21, 2009 considered claims 1, 3-6, 8-16, 18-20, and 22-35 each of which were rejected under 35 U.S.C. § 103(a) as being unpatentable over Cynerman and Jerger (U.S. Pat. No.: 6,321,334).

The current amendments have been made to better clarify the invention and to better support Applicant's arguments below. These arguments are similar to those that were discussed during the in person interview conducted before filing the last response, and included in the last response. Although the claims have been amended substantially, these amendments should not significantly affect the scope of the claims. The majority of the amendments have been made to provide correct and more concise language than was presented previously, particularly with respect to the dependent claims.

Each of the independent claims has been amended to better emphasize that the build process involves building a project that includes one or more build entities (e.g. source files), and that this building includes compiling at least one of the these build entities. The reason for this amendment will be addressed below, but in general is to limit the examiner's interpretation of the build process. The remaining amendments to the independent claims have only been made to clarify the limitations and to remove unnecessary language.

In the previous response, Applicant argued that the Ant tool is only a scripting tool for automating the build process, but is not the actual "build" process. Ant is simply used to construct build scripts. The reason for creating a script is so that the user does not have to reenter commands into the command prompt each time he desires to compile his code. Ant does not build (compile, link, etc.) code. It only invokes other executables that perform the build. Specifically, Ant is designed to be used with the JDK (Java Development Kit). The JDK is the software that includes the executables that perform the build process.

The reason why this distinction is important is that in rejecting claims 1 and 11, the examiner is equating Ant with the claimed "build process processor." The amendments make it clear that the build process processor of the claims is the component that is actually building the project. Specifically, the amendments state that the build process processor compiles at least one of the build entities. Ant cannot do this because it is only a scripting tool that invokes command line commands. In other words, Ant only calls the Javac command on a source file to cause the

JDK to compile the source file. Ant has no other involvement in the build process other than to invoke these types of commands. For example, please refer to the list of commands that can be included in the Ant script that are shown in the table on page 1 and 2 of the Ant reference. As further proof of this, please also refer to the example Ant script (or XML file) that is shown on page 3. Please note that in this example, on the 15th line, the javac command is listed. Consequently, when this Ant script is invoked, the commands listed within the script (including javac as well as deltree, mkdir, etc.) are invoked just as if the developer had typed them into the command prompt himself.

To further emphasize that Ant is only a scripting tool, Applicant refers the examiner to his own arguments in the Response to Arguments section of the Office Action. In the first portion of this section, the examiner argues that Ant is "a build process which is construed to encompass compiling, linking, and the like." To support this assertion, the examiner argues that Ant is the same as the make tool. Applicant completely agrees that Ant is similar to the make tool. However, the make tool, like Ant, does not compile, link, or perform any other role in the actual building of a project. As is stated in the GNU manual describing the make utility which is available at http://www.gnu.org/software/make/manual/make.html#Overview, "The make utility automatically determines which pieces of a large program need to be recompiled, ***and issues commands to recompile them.*** In other words, as you are designing a project, you will modify various source files, but generally not all of them. Make is a utility that will figure out which files have been modified, and thus need to be recompiled, for you. It will then issue the commands to recompile these files. This is just what Ant does, but it is XML based and specific to Java. Therefore, the examiner's argument that Ant is a build process that compiles because it is similar to the make utility is completely erroneous.

In summary, the reason why the amendments specify that the build process includes compiling is to distinguish from Ant's use of build process. Ant does call itself a build process, but the part that Ant plays in the build process is simply to invoke commands to cause the build to be performed.

As already stated, the actual build is performed by the JDK. The present invention would be applicable to how the JDK performs the build. For example, when the present invention commences a build, it would access the policy files to determine which level of trust is assigned to each entity that will be involved in the build. The lowest level of trust of any of the entities

would be selected as the level of trust under which the build of the project will be executed. This means that while the entities are compiled, linked, etc. during the build, the build will be limited to a certain level of access (or trust). Ant does not disclose anything similar to this. It is true that the developer can specify certain parameters for the various commands that can be executed by the JDK. The examiner makes specific reference to the include and exclude options of the javac command. However, as is stated in the Ant reference, the include and exclude options only specify which files to include or exclude from compilation. The specific example on page 5 shows that the developer wishes to include all java files except Script.java in the directory or any subdirectory of the srcdir specified in the Ant script. Specifying what files are included or excluded is not similar to determining what level of trust the entire build will execute under.

The examiner also argues that a build that is executed with a trusted permission level is the same as an Ant script that does not have any include or exclude statements, or that a build that is executed with a semi-trusted permission level is the same as an Ant script that has include or exclude statements. This makes no sense. If an Ant script contains no include or exclude statements, then all the .java files in the specified srcdir will be compiled, whereas if an Ant script contains include or exclude statements, only the .java files that match the statements will be compiled. This has nothing to do with a level of trust that is applied to a project being built. In particular, the present invention is not about what files are included or excluded from the build process, but is about what permission or access level the build will have while it is executing. For example, using the present invention, if you downloaded a source file from the internet, added it to your project, and then attempted to build it, the build of the entire project would fail because the level of trust for the downloaded source file would be untrusted. The reason for this is that generally you cannot trust that a source file you download from the internet is safe. Ant has absolutely nothing to do with anything like this.

Because Ant itself does not build projects, and only tangentially relates to the actual build process because it invokes commands of the JDK, Ant cannot teach or suggest the limitations of the claims. Specifically, Ant itself cannot teach "a build process processor that executes to build a project that includes one or more build entities, wherein building the project includes compiling at least one of the one or more build entities." Ant is not a processor that executes to build a project. It is admitted that the JDK would be a build processor; however, this is not how

the examiner is rejecting the claims, and the invention alone is not simply about building projects.

Ant also cannot teach "a policy component that is accessed by the build process processor before building the project to determine a level of trust within which the build process executes." The arguments above address how this limitation is not taught. In short, neither Ant nor the JDK access a policy to determine what the level of trust is for every single build entity that is involved in the build of a project. As will be discussed below, Jerger also does not disclose this limitation.

Finally, Ant also cannot teach "wherein the level of trust within which the build process executes is determined by analyzing the levels of trust associated with each of the one or more build entities, and selecting the lowest level of trust of all involved build entities ...." Ant discloses nothing even remotely similar to selecting which of the build entities has the lowest level of trust assigned to it or using this level of trust as the level of trust under which the entire build will be executed. As will be discussed below, Jerger also does not disclose this limitation.

The same arguments apply to the similar limitations of the other independent claims.

Now with reference to the Jerger reference, because Ant is only remotely related to the novel aspects of the present invention, and because Jerger is in no way related to building projects, it is unreasonable to combine the references as the examiner has done.

Jerger is related to surfing the web, not building software. Jerger assigns security zones to different websites which is a common feature found in Internet Explorer. For example, if you click on the tools menu, then Internet Options, and then the Security tab of the pop-up window, you will see what Jerger is directed to. Various different security zones are provided. Within these zones, you can specify websites (which you can see by clicking on the sites button). Websites that are listed in these zones are governed by different rules (which you can configure by clicking on the custom level button). As can be seen, this has nothing to do with building a project. The websites listed are not part of a project that will be built together and governed by a specified level of trust during the build. In short, Jerger discloses nothing that would lead one of ordinary skill in the art to modify Ant to yield the present invention. It is also unlikely that one of skill in the art would look to web surfing technology when designing improvements to how software is built. The examiner's reason for why these two references would be combined is that "One would have been motivated to do so to secure the build process by automatically administering the decision to grant or deny permission to specific build entities as suggested by

Jerger." OA, pg. 6. This reason is not valid. The invention does not grant or deny permission to specific build entities. In contrast, it looks at the lowest level of trust assigned to a build entity in the project being built, and then assigns that level of trust to the entire build. Therefore, the examiner's rationale for combining the references is irrelevant to the invention.

Jerger does not teach the limitations that the examiner is suggesting it does. In the rejection to claim 1, the examiner states that "Cynerman does not explicitly disclose determining one or more levels of trust within which the build process operates." In the current claims, this limitation is defined as selecting the lowest level of trust of any of the build entities involved. This is the fundamental aspect of the invention. However, to reject it, the examiner argues that Jerger's application of security zones to individual websites teaches this aspect. As already addressed, Jerger has nothing to do with selecting a permission level or level of trust as is claimed (i.e. by determining the lowest level assigned to any of the build entities). Jerger cannot teach this because it simply applies a security zone's restrictions to a single website when the user visits the website. Multiple websites are not grouped into a project that is built. Websites cannot be built. Therefore, Jerger is unrelated to the invention and fails to teach or suggest any of the limitations whether alone or in combination with the Ant reference.

In summary, the examiner is not interpreting the claim language correctly. The invention is directed to applying a permission level to a build that is determined by finding the lowest assigned level of trust of any of the build entities involved. The current amendments have been made to better clarify these aspects. The cited art does not relate to these aspects, and therefore fails to teach or suggest each limitation of the independent claims. Applicant therefore respectfully requests that the rejections be withdrawn.

In view of the foregoing, Applicant respectfully submits that the other rejections to the claims are now moot and do not, therefore, need to be addressed individually at this time. In the event that the Examiner finds remaining impediment to a prompt allowance of this application that may be clarified through a telephone interview, the Examiner is requested to contact the undersigned attorney at (801) 322-8427.

Dated this 19th day of February, 2010.

Respectfully submitted,

/BRIAN D. TUCKER/

RICK D. NYDEGGER
Registration No. 28,651
BRIAN D. TUCKER
Registration No. 61,550
Attorneys for Applicant
Customer No. 47973

RDN:BDT
2704123_1